การเขียนโปรแกรมเรสซิเดนต์ตอนจบ

■ จเร เลิศสุดวิชัย

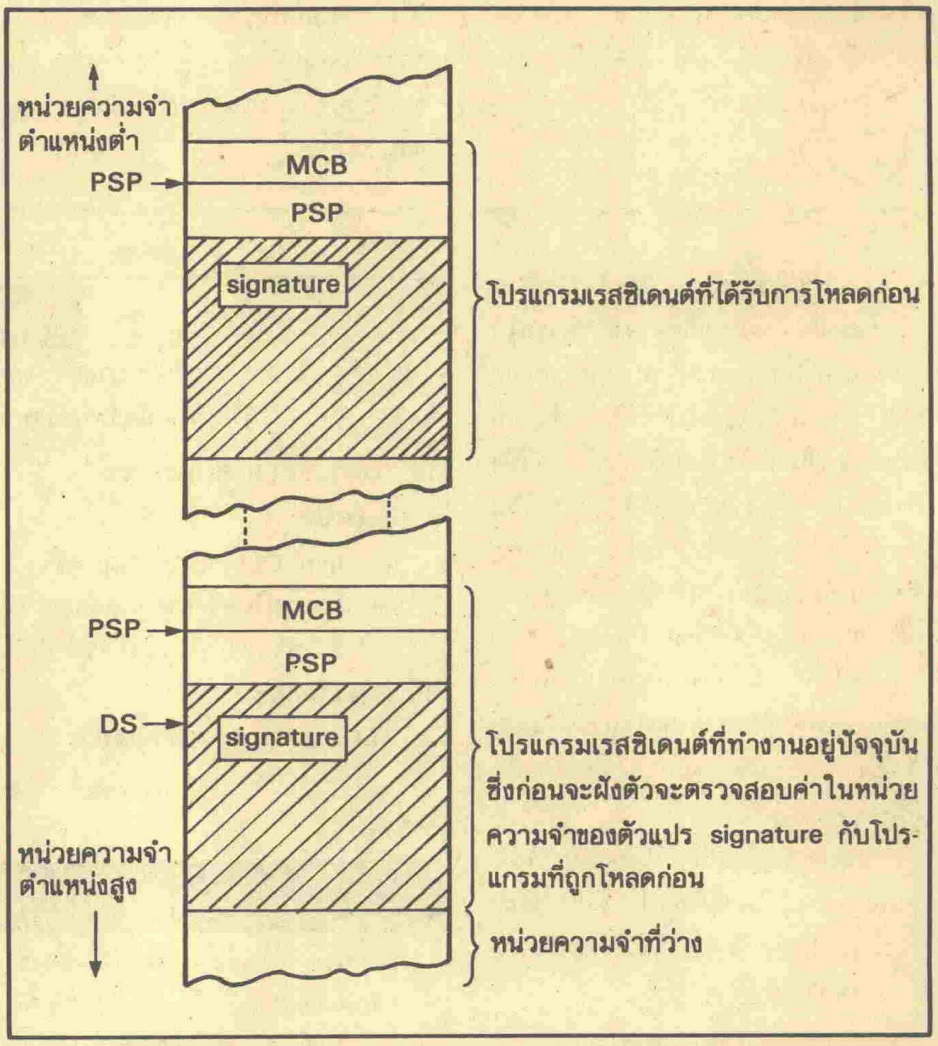
ในตอนที่แล้วได้อธิบายถึงขั้นตอนการเขียนโปรแกรมเรสซิเดนต์ พร้อมทั้งเทคนิคและความรู้ที่ต้องใช้ประกอบอย่างละเอียด รวมทั้งได้ลงโปรแกรมและฟังก์ชันที่ช่วยจัดการขั้นตอนการติดตั้งโปรแกรม การปลุกโปรแกรมเรสซิเดนต์และตัวอย่างป๊อบอัพรูทีน ซึ่งเขียนด้วยเทอร์โบซีไปแล้ว ในตอนสุดท้ายนี้ก็จะเป็นโปรแกรมที่เขียนด้วยไมโครซอฟต์ซี ซึ่งหลักการทำงานของโปรแกรมก็จะเหมือนเดิม

สำหรับการใช้ไมโครซอฟต์ซี เขียนโปรแกรมเรสซิเดนต์นั้นจำเป็นต้องอาศัยฟังก์ชันบางส่วนที่เขียนด้วยแอสแซมบลีช่วยในการทำงาน เช่น การเปลี่ยนค่าในรีจิสเตอร์ต่าง ๆ ดังนั้นโปรแกรมที่ลงพิมพ์ให้ในตอนนี้จึงแบ่งออกเป็น 2 ส่วนคือ ส่วนที่เป็นภาษาซีและส่วนที่เป็นแอสแซมบลี

การตรวจสอบและการป้องกันการโหลดโปรแกรมซ้ำ

วิธีการตรวจสอบโปรแกรม เพื่อป้องกันการโหลดโปรแกรมซ้ำนั้นได้กล่าวไว้ในตอนที่แล้ว ซึ่งมีด้วยกัน 2 วิธี ซึ่งในโปรแกรมของตอนที่แล้วใช้วิธีการเขียนหรือฝากข้อความหรือชื่อโปรแกรมไว้ที่อินเตอร์รัพต์เวกเตอร์ที่ว่างอยู่ (60h-67h) แต่สำหรับในตอนนี้จะใช้วิธีการตรวจสอบวิธีที่ 2 คือ การค้นหาชื่อหรือข้อความของโปรแกรมจาก PSP ของโปรแกรมที่ถูกโหลดลงหน่วยความจำโดยดอส ซึ่งวิธีการค้นหาจะเริ่มจาก MCB แรก (memory control blocks) โดยอาศัยฟังก์ชันของดอสคือ ฟังก์ชันที่ 52h (รายละเอียดดูในตอนที่แล้ว) ไล่ดู PSP ของแต่ละโปรแกรมในหน่วยความจำของดอสจนถึง PSP ของตัวโปรแกรมเองที่กำลังทำงานอยู่หรือจนกระทั่งสิ้นสุด MCB ในการตรวจสอบแต่ละ PSP นั้นจะทำการเปรียบเทียบชื่อ (Signature) ของโปรแกรม ซึ่งถูกกำหนดไว้เป็นตัวแปรโกลบอลของโปรแกรมที่ทำงานอยู่ขณะนั้นกับตำแหน่งหน่วยความจำที่ห่างจาก PSP หรือ DS เท่า ๆ กันของโปรแกรมเรสซิเดนต์ซึ่งฝังตัวอยู่ในหน่วยความจำมาก่อนแล้ว โดยตำแหน่งหน่วยความจำของข้อความนี้หาได้จาก PSP, ดาต้าเซกเมนต์ (Data Segment) และออฟเซตของตัวแปร Signature

ซึ่งรายละเอียดนั้นดูได้จากรูปที่ 1 และในฟังก์ชัน resident การตรวจสอบนั้น



รูปที่ 1 แสดงตำแหน่งที่อยู่ของตัวแปร Signature ในโปรแกรมเรสซิเดนต์ที่ถูกโหลดก่อน และโปรแกรมเรสซิเดนต์ที่ทำงานอยู่ปัจจุบัน

ถ้าปรากฏว่าค่าในตัวแปร Signature กับตำแหน่งในหน่วยความจำของโปรแกรมที่โหลดก่อนนั้นมีค่าเหมือนกันหมด แสดงว่าโปรแกรมเรสซิเดนต์ได้รับการติดตั้งก่อนแล้ว แต่ถ้าค้นหาจนกระทั่งพบ PSP ของตัวโปรแกรมตัวเองหรือจนสิ้นสุด MCB ซึ่งหมายถึงโปรแกรมเรสซิเดนต์ยังไม่เคยติดตั้งหรือฝังตัวมาก่อนเลย

ฟังก์ชันการอ่านคีย์

ฟังก์ชันการอ่านค่าคีย์ (รวมทั้งฟังก์ชัน kbhit ด้วย) ของไมโครซอฟต์ก็เช่นเดียวกับเทอร์โบซีคือ เป็นฟังก์ชันที่เรียกใช้ฟังก์ชันของดอสที่ต่ำกว่า 0ch ซึ่งไม่สามารถเรียกใช้ในโปรแกรมเรสซิเดนต์ได้ จึงได้ทำฟังก์ชัน get_char สำหรับอ่านคีย์ และฟังก์ชันสำหรับตรวจสอบการกดคีย์ keyhit ขึ้นใหม่ ซึ่งฟังก์ชัน get_char นั้นจะเรียกผ่านทางฟังก์ชัน _bios_keybrd แต่มีข้อควรระวังอันหนึ่งก็คือ ฟังก์ชัน _bio_keybrd (รวมทั้งฟังก์ชัน bioskey ของเทอร์โบซี) ไม่สามารถตรวจสอบการกด control-break ได้ สำหรับฟังก์ชันตรวจสอบการกดคีย์คือฟังก์ชัน keyhit นั้นจำเป็นต้องเขียนด้วยแอสเซมบลี เพราะว่าฟังก์ชัน int 86 ของไมโครซอฟต์นั้น ไม่สามารถจะรายงานค่าของแฟล็กศูนย์ (Zero flag) ซึ่งต้องใช้สำหรับการตรวจสอบการกดคีย์ได้

ฟังก์ชันการอ่าน, การกำหนด DTA, ฟังก์ชัน peek, peekb และฟังก์ชัน poke

สำหรับฟังก์ชัน getdta และ setdta ในไมโครซอฟต์ซีไม่มีให้ใช้จึงต้องเขียนขึ้นมาใหม่ โดยจะเรียกใช้ฟังก์ชันของดอสผ่านทางฟังก์ชัน int 86 อีกที ส่วนฟังก์ชัน peek, peekb และ poke ก็เช่นกัน ต้องเขียนขึ้นใหม่โดยอาศัยเรื่องพอยน์เตอร์และมาโคร FP_SEG และ FP_OFF ช่วย

อินเตอร์รัพต์ของ divide-by-zero

ดังที่เคยกล่าวมาแล้วในตอนที่แล้วว่าอินเตอร์รัพต์ของ divide-by-zero จะถูกเปลี่ยนค่าในเริ่มโปรแกรม และถ้าโปรแกรมจบลงที่ฟังก์ชัน main ค่าของอินเตอร์รัพต์นี้จึงจะถูกคืนกลับ แต่สำหรับโปรแกรมเรสซิเดนต์นั้นโปรแกรมจะจบลงที่ฟังก์ชัน keep ซึ่งในกรณีนี้อินเตอร์รัพต์ของ devide-by-zero จะไม่ถูกเปลี่ยนกลับซึ่งอาจจะทำให้ปัญหาขึ้นได้ ในไมโครซอฟต์จะมีฟังก์ชันในส่วนของ start-up code คือ ฟังก์ชัน _ctermsub การทำงานของฟังก์ชันนี้จะคืนค่าของอินเตอร์รัพต์นี้แก่ดอส โดยก่อนจะเรียกฟังก์ชัน keep จึงจะต้องเรียกฟังก์ชันนี้ก่อน

ฟังก์ชันการแสดงผลบนจอภาพ

สำหรับการแสดงผลบนจอนั้น จะใช้วิธีเขียนลงไปในส่วนหน่วยความจำแสดงผลโดยตรง ซึ่งฟังก์ชันจะประกอบด้วยกัน 2 ส่วน คือ ส่วนที่เป็นภาษาซีและส่วนที่เป็นภาษาแอสเซมบลี ฟังก์ชันการแสดงผลที่ใช้จะต้องตรวจสอบชนิดของจอภาพก่อนเพื่อหาตำแหน่งของหน่วยความจำแสดงผล ซึ่งจะต้องเรียกฟังก์ชัน fast-display ในเริ่มต้นของโปรแกรมเรสซิเดนต์คือในฟังก์ชัน resint ก่อน

โปรแกรมตัวอย่างและการใช้งาน

โปรแกรมที่พิมพ์ให้มานั้นจะประกอบไฟล์ช่วยการทำงานโปรแกรมเรสซิเดนต์ซึ่งประกอบด้วยไฟล์ต่อไปนี้คือ

TSR.C เป็นไฟล์หลักในการควบคุมการหาขนาดโปรแกรม และการกำหนดคีย์พิเศษ (Hot key) ฟังก์ชัน main และฟังก์ชันที่ไม่มีในไลบารี่ของไมโครซอฟซี ซึ่งได้แก่ฟังก์ชัน การอ่าน-กำหนด DTA (getdta และ setdta) ฟังก์ชันการอ่าน-กำหนดค่าให้หน่วยความจำ (peek, peekb และ poke) และฟังก์ชันการอ่านคีย์ซึ่งเขียนใหม่โดยผ่านทางบริการของไบออสอีกทีหนึ่ง

RESIDENT.C เป็นไฟล์ที่ประกอบด้วยฟังก์ชันช่วยในการติดตั้งและการทำงานของโปรแกรมเรสซิเดนต์

TSRASM.ASM เป็นไฟล์ที่ประกอบด้วยฟังก์ชันที่ไม่สามารถเขียนด้วยภาษาซีได้ ซึ่งจะเป็นฟังก์ชันที่มีการเปลี่ยนแปลงค่าในรีจิสเตอร์ต่าง ๆ

VIDEOC.C เป็นไฟล์ที่มีฟังก์ชันช่วยในการแสดงผล ซึ่งการแสดงผลของฟังก์ชันเหล่านี้จะทำกับหน่วยความแสดงผล (Video memory) โดยตรง

VIDEOA.ASM เป็นไฟล์ที่มีฟังก์ชันช่วยการทำงานของฟังก์ชันในไฟล์ VIDEO C.C แต่เขียนด้วยภาษาแอสเซมบลี

CLOCK.C เป็นไฟล์ที่มีฟังก์ชันป๊อบ-อัพรูทีน ตัวอย่างซึ่งจะถูกเรียกใช้งานเมื่อโปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมา โดยจะแสดงเวลาและวันบนจอภาพ

ตัวอย่างโปรแกรมจะเป็นการแสดงวันและเวลาบนจอภาพ เมื่อกด ATL พร้อมกับ "C" และการทำงานที่เปลี่ยนไปสำหรับการถอนโปรแกรมและหยุดโปรแกรมชั่วคราว จะทำได้โดยผ่านทางการกดคีย์ขณะที่แสดงเวลาอยู่ การถอนโปรแกรมจะใช้คีย์ "U" ส่วนการหยุดโปรแกรมชั่วคราวจะใช้คีย์ "S" และสำหรับการเรียกโปรแกรมให้ทำงานดังเดิม เมื่อหยุดโปรแกรม

ชั่วคราวจะทำได้ โดยการเรียกชื่อโปรแกรมเมื่ออยู่บนดอสอีกครั้ง

การทดสอบและข้อควรคำนึงในการเขียนโปรแกรมเรสซิเดนต์

การทดสอบโปรแกรมเรสซิเดนต์นั้นครั้งแรกควรจะทดสอบส่วนที่เป็นป๊อบอัพรูทีนก่อน โดยทำงานในลักษณะของโปรแกรมธรรมดา (Trasient Program) แก้ไขข้อผิดพลาดให้ถูกต้องก่อน จึงนำมารวมกับฟังก์ชันที่ช่วยในการทำเรสซิเดนต์ แล้วจึงทดสอบในลักษณะที่เป็นโปรแกรมเรสซิเดนต์ อย่างไรก็ตามการทดสอบในขั้นแรก (ทดสอบแบบโปรแกรมธรรมดา) จนไม่พบข้อผิดพลาดนั้น ไม่ได้เป็นการประกันว่าเมื่อทำเป็นโปรแกรมเรสซิเดนต์แล้วไม่มีข้อผิดพลาดเกิดขึ้นอีก ทั้งนี้เพราะว่าลักษณะของโปรแกรมเรสซิเดนต์กับโปรแกรมธรรมดานั้นไม่เหมือนกัน เพราะโปรแกรมเรสซิเดนต์นั้นในการทำงานแต่ละครั้งจะใช้โปรแกรมและข้อมูลชุดเดิมเสมอ

การเขียนโปรแกรมเรสซิเดนต์ที่ต้องการใช้ทำงานได้กว้าง หรือมีฟังก์ชันการทำงานได้มากขึ้นนั้นจะต้องอาศัยความรู้ในเรื่อง ฮาร์ดแวร์, โอเอส (MSDOS) และความรู้ทางด้านซอฟต์แวร์ต่าง ๆ ที่ต้องใช้มากขึ้นตามไปด้วย ซึ่งจะขอยกให้ผู้อ่านที่จะเขียนโปรแกรมเรสซิเดนต์ไปศึกษาเอง ยกตัวอย่างเช่น การทำงานอินเตอร์รัพต์ของไบออสบนเครื่อง PC หรือ AT คอมแพตติเบิ้ลนั้น ฟังก์ชันบริการอินเตอร์รัพต์บางอย่างไม่ได้ตรง หรือเหมือนกับของ IBM ทั้งหมด โดยการจะมีการส่งผ่านทางรีจิสเตอร์นอกเหนือไปจากของเดิม ดังนั้นเขียนโปรแกรมหรือฟังก์ชันที่เข้าไปแทนที่อินเตอร์รัพต์ส่วนนี้ก็จะต้องรักษาในรีจิสเตอร์ต่าง ๆ เอาไว้ด้วย

มาถึงตอนนี้หวังว่าผู้อ่านคงได้รับความรู้หลักการทำงานของโปรแกรมเรสซิเดนต์ และหวังว่าคงจะได้นำไปใช้ประโยชน์ในการพัฒนางานหรือโปรแกรมที่เขียนอยู่ให้มีความสามารถเพิ่มขึ้น และในฉบับต่อไปมุมภาษาซีก็จะหาบทความที่มีประโยชน์มาเสนอให้กับท่านผู้อ่านอีกเช่นเคย

เอกสารอ้างอิง

1. IBM "Technical Reference Manual For IBM PC-XT", 1983

2. IBM "Technical Reference Manual For IBM PC-AT", 1983

3. John Angermeyer and Kevin Jaeger "MSDOS Developer's Guide" Howard W.Sans & Co., Indianapolis USA. 1986

4. Stephen K, O'Brien "Turbo Pascal Advanced Programer's Guide", BOLAND OSBORNE, 1988

5. AL Stevens "Writing terminate-and-Stay-Resident Programs", Compute Language, February and March, 1988

6. บุญภพ ลิ่มมโนกุล "การจัดการหน่วยความจำของดอส", วารสารไมโครคอมพิวเตอร์ฉบับ 50, บริษัทซีเอ็ดยูเคชั่น จำกัด, 1988

7. สุรศักดิ์ สงวนพงษ์ "เขียนโปรแกรม COM และ EXE ให้ถูกวิธี", วารสารไมโครคอมพิวเตอร์ ฉบับ 32, บริษัทซีเอ็ดยูเคชั่น จำกัด 1987

```
/*------------- tsr.h --------------*/
void resinit (void);
void resident_psp (void);
void interrupted_psp (void),
int  resident (char *);
void terminate (void);
void restart (void);
void suspend (void);
int  get_char (void);
char far *getdta (void);
void setdta (char far *);
unsigned peek (unsigned ,unsigned);
char peekb (unsigned ,unsigned);
void poke (unsigned ,unsigned ,unsigned);
void popup (void);
void openfiles (void);
void closefiles (void);
unsigned getsp (void);
int  keyhit (void);
int  keyrdy (void);
```

รูปที่ 2 เฮดเดอร์ไฟล์ของไฟล์ TSR.C

รูปที่ 3 ไฟล์ TSR.C

```
/*----------------- tsr.c -----------------*/
#include <dos.h>
#include <stdio.h>
#include <string.h>
#includa <bios.h>
#include "tsr.h"
#pragma check_stack (off)

static union REGS rg;
static struct SREGS sg;

/*---------------- Modify this for each TSR -------------*/
unsigned heaplen  = 128;              /* Define heap length     */
unsigned scancode = 46;               /* Hot key scan code  'c' */
unsigned keymask  = 8;                /* Hot key status mask ALT*/
char signature [] = "CLOCK resident Program";  /* TSR signature*/

/*---------- main function for the TSR program ---------*/
void main (void)
{
   int r;
   if ((r = resident (signature)) == 0)
      {
      printf ("\nResident clock is loaded");
      resinit ();
      }
   printf ("\nClock is %s ",(r==1 ? "already resident" : " restart"));
}

/*------------ get a keyboard chararter -----------*/
int get_char (void)
{
   int c;

   while (keyhit() == 0)

   c = _bios_keybrd (_KEYBRD_READ);
   return ( (c & 0xff) == 0) ? (((c >> 8) & 0xff) + 128) : (c & 0xff);
}

/*------------- set the Disk Tranfer Address -----------*/
void setdta (char far *dta)
{
   rg.h.ah = 0x1a;
   rg.x.dx = FP_OFF (dta);
   sg.ds   = FP_SEG (dta);
   int86x (0x21,&rg,&rg,&sg);
}

/*-------------- get the Disk Tranfer Address -----------*/
char far * getdta (void)
{
   static char far *dta;

   rg.h.ah = 0x2f;
   int86x (0x21,&rg,&rg,&sg);
   FP_SEG (dta) = sg.es;
   FP_OFF (dta) = rg.x.bx;
   return (dta);
}

/*--------------- peek, peekb ,poke --------------------*/
unsigned peek (unsigned s, unsigned o)
{
   static unsigned far *ip;

   FP_SEG (ip) = s;
   FP_OFF (ip) = o;
   return (*ip);
}

char peekb (unsigned s, unsigned o)
{
   static unsigned far *ip;

   FP_SEG (ip) = s;
   FP_OFF (ip) = o;
   return (*ip);
}

void poke (unsigned s, unsigned o, unsigned c)
{
   static unsigned far *ip;

   FP_SEG (ip) = s;
   FP_OFF (ip) = o;
   *ip = c;
}
```

รูปที่ 4 ไฟล์ TSRASM.ASM

```
;------------------------ tsrasm.asm ------------------------
        dosseg
        .model small
        .data

        public _running
        extrn  _myss       : word
        extrn  _stack      : word
        extrn  _suspending : word
        extrn  _olddisk    : word
        extrn  _diskflag   : word

intss   dw     ?
intsp   dw     ?
_running db    ?

        .code
extrn   _popup_tsr : near

public  _getsp
;-------- return the stack pointer ---------
;       stackpointer = getsp()
;-------------------------------------------
_getsp  proc   near
        mov    ax,sp
        ret
_getsp  endp

public  _keyhit
;-------- test  for a keystroke, return 1 if key ready, 0 if not
;         k = keyhit();
;-------------------------------------------
_keyhit proc   near
        mov    ah, 1        ; BIOS read key status
        int    16h
        mov    ah, 1        ; in case of ctrl-break
        jnz    keyret       ; if a key is ready
        int    28h          ; else, call int 28 to allow other TSRs in *
        mov    ax, 0        ; and return false
keyret: ret
_keyhit endp

public  _dores
;-------- swap the stack and call the popup function
;       dores ();
;-------------------------------------------
_dores  proc   near
        cmp    _suspending, 0
        jne    resret       ; don't pop up if the TSR is suspended
        mov    _running, 1 ; prevent reentrant TSRs
        cli
        mov    intss, ss    ; save the interrupted stack
        mov    intsp, sp
        mov    ss, _myss    ; set the TSR's stack
        mov    sp, _stack
        sti
        call   _popup_tsr   ; pop up the utility
        cli
        mov    ss, intss    ; restore the interrupted stack
        mov    sp, intsp
        sti
        mov    _running, 0
resret: ret
_dores  endp

public  _newdisk
;-------- 0x13 (disk) BIOS interrupt Service Routine
;-------------------------------------------
_newdisk proc far
        push   ds
        push   bp
        mov    bp,DGROUP    ; set ds to TSR's data segment
        mov    ds,bp
        inc    word ptr DGROUP:_diskflag
        pushf
        call   dword ptr DGROUP:_olddisk  ; chain to old disk interrupt

        dec    word ptr DGROUP:_diskflag  ; reset disk in process flag
        pop    bp
        pop    ds
        ret    2            ; ret 2 passes on the flags
_newdisk endp

end
```

มุมภาษา C

รูปที่ 6 ไฟล์ RESIDENT.C

```
/* ------------------ resident.c ---------------- */
/*                Microsoft - C compiler         */
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include "tsr.h"
#pragma check_stack (off)
#pragma intrinsic (inp, outp)

static union REGS rg;
static struct SREGS seg;

/* ---------- vectors ---------- */
#define KEYBD 0x09
#define TIMER 0x1c
#define DISK  0x13
#define INT28 0x28
#define CRIT  0x24
#define CTRLC 0x23
#define CTRLBRK 0x1b

/* ---------- interrupt vector chains ---------- */
static void (interrupt far *oldbreak) (void);
static void (interrupt far *oldctrlc) (void);
static void (interrupt far *oldtimer) (void);
static void (interrupt far *old28) (void);
static void (interrupt far *oldkb) (void);
static void (interrupt far *oldcrit) (void);
void (interrupt far *olddisk) (void);

/* ---------- isr for the tst ---------- */
static void interrupt far newtimer (void);
static void interrupt far new28 (void);
static void interrupt far newkb (void);
static void interrupt far newcrit (int,int,int,int,int,int,int,int,int,int);
static void interrupt far newbreak (void);
extern void interrupt far newdisk (void);

extern unsigned heaplen;        /* TSR's heap size          */
static unsigned sizeprogram;    /* TSR's program size       */
static unsigned dosseg,         /* DOS segment address      */
```

```
static char far *dosbusy;       /* points to InDOS flag     */
static char far *mydta;         /* TSR's DTA                */
static unsigned intpsp;         /* Interrupt PSP address    */
static unsigned psps[2];  /* Table of DOS PSP addresses */
static int pspctr;              /* # of DOS PSP addresses   */
static int popflg;              /* Hot key pressed flag     */
unsigned myss;                  /* TSR's stack segment      */
unsigned stack;                 /* TSR's stacd pointer      */
int suspending;                 /* suspend/resume flag      */
int terminating;                /* terminate flag           */
int diskflag;                   /* Disk BIOS busy flag      */

/* ---------- local prototypes ---------- */
static void resterm (void);
static void pspaddr (void);
void dores (void);
void _ctermsub (void);
void popup_tsr (void);

/* ---------- establish & declare residency ---------- */
void resinit ()
{
   /* ----- Detect Screen for display ----*/
   fast_display ();

   /* ----- save TSR stack pointer ----- */
        stack = getsp ();
        segread (&seg);
        myss = seg.ss;

   /* ----- get address of DOS busy flag ----- */
        rg.h.ah = 0x34;
        int86x (0x21, &rg, &rg, &seg);
        dosseg = seg.es;
        FP_SEG (dosbusy) = seg.es;
        FP_OFF (dosbusy) = rg.x.bx;

   /* ----- get address of resident program's dta ----- */
        mydta = getdta ();

   /* ----- get addresses of PSP in DOS ----- */

        pspaddr ();
```

รูปที่ 5 (ต่อ)

```
        /* ----- get original vectors ----- */
        oldtimer = _dos_getvect (TIMER);
        old28    = _dos_getvect (INT28);
        oldkb    = _dos_getvect (KEYBD);
        olddisk  = _dos_getvect (DISK);

        /* ----- attach vectors to resident program ----- */
        _dos_setvect (TIMER, newtimer);
        _dos_setvect (INT28, new28);
        _dos_setvect (KEYBD, newkb);
        _dos_setvect (DISK, newdisk);

        /* ----- compute program size ----- */
        sizeprogram = (heaplen / 16) + ((myss + ((stack + 50) / 16)) - _psp)

        /* ----- restore zero divide interrupt vector ----- */
        _ctermsub ();

    /* ----- terminate and stay resident ----- */
        _dos_keep (0, sizeprogram);
}

/* ---------- break handler ---------- */
static void interrupt far newbreak ()
{
   return;
}

/* ---------- critical error ISR ---------- */
static void interrupt far newcrit (es, ds, di, si, bp, sp, bx, dx, cx, ax)
{
   ax = 0;
}

static int kbval;

/* ---------- keyboard ISR ---------- */
static void interrupt far newkb ()
{
   extern unsigned scancode,
                   keymask, running;

        if (scancode && inp (0x60) == scancode)
        {
        kbval = peekb (0,0x417) & keymask;
        if (!suspending && (kbval == keymask))
           {
           /* ----- reset the keyboard ----- */
           kbval = inp (0x61);
           outp (0x61, kbval | 0x80);
           outp (0x61, kbval);
           _disable ();
           outp (0x20, 0x20);
           _enable ();
           /* ----- set hotkey indicator ----- */
           if  (!running)
               popflg = 1;
           return;
           }
        }
        (*oldkb) ();
}

/* ---------- timer ISR ---------- */
static void interrupt far newtimer ()
{
   extern unsigned scancode, keymask, running;

   (*oldtimer) ();
   if (!scancode)
      {
      kbval = peekb (0,0x417) & keymask;
      if (!suspending && (kbval == keymask))
         if (!running)
            popflg = 1;
       }
   if (popflg && *dosbusy == 0)
      if (diskflag == 0)
         {
         outp (0x20, 0x20);
         popflg = 0;
         dores ();
         }
}
```

มุมภาษา C รูปที่ 6 (ต่อ)

```
/* ---------- INT28 ISR ---------- */
static void interrupt far new28 ()
{
   (*old28) ();
   if (popflg && *dosbusy != 0 )
      {
      popflg = 0,
      dores ();
      }
}

/* ---------- switch psp context interrupt to TSR ---------- */
void resident_psp ()
{
   int pp;

   /* save interrupted program's psp */
   intpsp = peek (dosseg, *psps);
   /* set resident program's psp     */
   for (pp = 0; pp < pspctr; pp++)
       poke (dosseg, psps[pp],_psp);
}

void interrupted_psp ()
{
   int pp;

   /* reset interrupted program's psp */
   for (pp = 0; pp < pspctr; pp++)
       poke (dosseg,psps [pp], intpsp);
}

void popup_ter ()
{
   static char far *intdta;         /* interrupt DTA            */
   static unsigned intsp, intss;    /* interrupt stack          */
   static unsigned char ctrl_break; /* Ctrl-Break setting       */

   oldcrit  = _dos_getvect (CRIT);    /* redirect critical error   */
   oldbreak = _dos_getvect (CTRLBRK);
   oldctrlc = _dos_getvect (CTRLC);
   _dos_setvect (CRIT, newcrit);
   _dos_setvect (CTRLBRK, newbreak);
   _dos_setvect (CTRLC, newbreak);
   rg.x.ax = 0x3300;                /* get ctrl break setting    */
   int86 (0x21, &rg, &rg);
   ctrl_break = rg.h.dl;
   rg.x.ax = 0x3301;                /* turn off ctrl break logic */
   rg.h.dl = 0;
   int86 (0x21, &rg, &rg);
   intdta = getdta ();               /* get interrupted dta       */
   setdta (mydta);                   /* set resident dta          */
   resident_psp ();                  /* swap psps                 */
   popup ();                         /* execute resident program  */
   interrupted_psp ();               /* reset interrupted psp     */
   setdta (intdta);                  /* reset interrupted dta     */
   _dos_setvect (CRIT, oldcrit);     /* reset critical error      */
   _dos_setvect (CTRLBRK, oldbreak);
   _dos_setvect (CTRLC, oldctrlc);
   rg.x.ax = 0x3301;                /* reset ctrl break          */
   rg.h.dl = ctrl_break;
   int86 (0x21, &rg, &rg);
   if  (terminating)
      terminate ();
}

/* --------- test to see if the TSR is aleady resident ---------- */
int resident (signature)
char *signature;
{
   char *sg;
   unsigned df, mcbseg, blkseg;

   segread (&seg);
   df = seg.ds - _psp;

   /* get the seg addr of 1st DOS MCB */
   rg.h.ah = 0x52;
   intdosx (&rg, &rg, &seg);
   mcbseg = peek (seg.es, rg.x.bx - 2);

   /* walk thru chain, srch for TSR */
   while (peekb (mcbseg, 0) == 0x4d)
        {
        blkseg = peek (mcbseg, 1);
        if (peek (blkseg, 0) == 0x20cd)
```

รูปที่ 5 (ต่อ)

```
            {
            if  (blkseg == _psp)     /* this is a psp */
                break;               /* if the transient copy */
            for (sg = signature; *sg; sg++)
                if (*sg != peekb (blkseg + df,  (unsigned)sg))
                   break;

            /* if TSR is already resident */
            if (!*sg)
               {
               if (peek (blkseg + df,  (unsigned) & suspending))
                  {
                  poke (blkseg + df,  (unsigned) & suspending, 0);
                  return 2;      /* unsigned a suspended TSR */
                  }
               return 1;
               }
            }
         mcbseg += peek (mcbseg, 3) + 1;
         }
   return 0;
}

/* ---------- psp address ---------- */
static void pspaddr ()
{
   unsigned adr = 0;
   unsigned enddos;              /* offset to the end of DOS */

   rg.h.ah = 0x52;               /* find the end of the DOS segment */
   int86x (0x21, &rg, &rg, &seg);
   enddos = peek (seg.es,rg.x.bx - 2);
   /* search for match on the psp in dos */
   while  (pspctr < 2 && (unsigned) ( (dosseg << 4) + adr)
          <  (enddos << 4))
      {
      if  (peek (dosseg,adr) == _psp)
         {
         rg.h.ah = 0x50;
         rg.x.bx = _psp + 1;
         int86 (0x21, &rg, &rg);
         if (peek (dosseg,adr) == _psp + 1)
            {
            psps[pspctr++] = adr;
            }
         rg.h.ah = 0x50;         /* reset the originnal psp */
         rg.x.bx = _psp;
         int86 (0x21, &rg, &rg);
         }
      adr++;
   }
}

/* ---------- terminate function ---------- */
static void resterm ()
{
   static unsigned far *env;

   _dos_setvect (TIMER, oldtimer); /* restore the interrupt vector */
   _dos_setvect (INT28, old28);
      _dos_setvect (KEYBD, oldkb);
   _dos_setvect (DISK, olddisk);
   FP_SEG (env) = _psp;          /* release memory owned by the TSR */
   FP_OFF (env) = 0x2c;
   _dos_freemem (*env);
   _dos_freemem (_psp);
}

/* ---------- terminate the resident program ---------- */
void terminate ()
{
   if  (_dos_getvect (DISK) ==  (void  (interrupt far *) ()) newdisk)
      if  (_dos_getvect (KEYBD) == newkb)
         if  (_dos_getvect (INT28) == new28)
            if  (_dos_getvect (TIMER) == newtimer)
               {
               resterm ();
               return;
               }
   terminating = 0;      /* another TSR is above, suspend */
   suspending = 1;
```

รูปที่ 6 เฮดเดอร์ไฟล์ของ VIDEOC.C

```
/*------------------------- video.h -------------------*/
#define NoDisplay       0x00
#define LowDisplay      0x07
#define HighDisplay     0x0f
#define UnderLineLow    0x01
#define UnderLineHihg   0x09
#define ReverseLow      0x70
#define ReverseHigh     0x7f
#define BlinkLow        0x87
#define BlinkHigh       0x8f
#define UnderBlinkLow   0x81
#define UnderBlinkHigh  0x89
#define RevBlinkLow     0xf0
#define RevBlinkHigh    0xf8
#define True            01
#define False           0

/* Control and status port address */
#define Mono_crt_status   0x3ba
#define Color_crt_status  0x3da
#define Mono_Seg     0xb000
#define Color_Seg    0xb800

typedef struct
{
    unsigned cursor_pos;        /* current possition on screen */
    unsigned stport;            /* CRT I/O port address        */
    unsigned char  char_attrib; /* cureen screen atrribute     */
}   SCREEN_DATA;

/*-----------------------------------------------------------------*/
void            fast_display ();
unsigned        f_getc (void);
void            f_gotoxy (char x,char y);
void            f_putc (unsigned char c);
void            f_puts (unsigned char  *str);
void            textattr (unsigned char attrib);
void            set_size_cursor (unsigned char top,unsigned char bottom);
void            cursor_on (void);
void            cursor_off (void);
extern void     fast_putc (char);
extern void     fast_puts (char);
extern int      fast_getc (void);
```

รูปที่ 7 ไฟล์ VIDEOC.C

```
/*------------------------- videoc.c ----------------------*/
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <dos.h>
#include <bios.h>
#include <ctype.h>
#include "video.h"
#pragma check_stack (off)

extern SCREEN_DATA   direct_video;
unsigned int  screenseg;

/*------------- find video display memory -----------*/
void fast_display ()
{
   union REGS  regs;

   int86(0x11,&regs,&regs);
   if ((regs.x.ax & 0x30)==0x30)
              {
      screenseg=Mono_Seg;
      direct_video.stport=Mono_crt_status;
              }
   else
              {
      screenseg=Color_Seg;
      direct_video.stport=Color_crt_status;
              }
   direct_video.cursor_pos=0;
   direct_video.char_attrib=LowDisplay;
}

/*----------- move position to write chararter to (x,y)----------*/
void f_gotoxy(char x,char y)
{
   direct_video.cursor_pos=((int)(y-1) * 80) +(int) (x-1) << 1;
}

/*-------------- change cursor size ------------------------*/
void set_size_cursor(unsigned char top,unsigned char bottom)
{
  unsigned  int  far *cursormode = (unsigned int far *) 0x004000601;
```

มุม ภาษา C

รูปที่ 7 (ต่อ)

```
    unsigned  int  far *videoport  = (unsigned int far *) 0x004000631;

    if (!screenseg==Mono_Seg)
       {
       top=top-4;
       bottom=bottom-4;
       }
    outp(*videoport,10);
    outp(*videoport+1,top);
    outp(*videoport,11);
    outp(*videoport+1,bottom);
    *cursormode=(top << 8) | bottom;
}

/*---------------- on cursor -----------------*/
void cursor_on()
{
  unsigned  int  far *cursormode = (unsigned int far *) (0x004000601);
  unsigned  int  far *videoport  = (unsigned int far *) (0x004000631);

  outp(*videoport,10)
  outp(*videoport+1,((unsigned char )(*cursormode >> 8)) & 0xdf );
  outp(*videoport,11);
  outp(*videoport+1,(unsigned char )*cursormode);
}

/*------------------of cursor -------------------*/
void cursor_off()
{
  unsigned  int  far *cursormode = (unsigned int far *) (0x004000601);
  unsigned  int  far *videoport  = (unsigned int far *) (0x004000631);

  outp(*videoport,10);
  outp(*videoport+1,((unsigned char )(*cursormode >> 8)) | 0x20 );
  outp(*videoport,11);
  outp(*videoport+1,(unsigned char )*cursormode);
}

/*---------------- set text attribute -------------------*/
void textattr (unsigned char attr)
{
  direct_video.char_attrib=attr;
}
```

```
/*---------- put a text attribute on position display---------*/
void f_puta (unsigned char attr)
{
        fast_puta (attr);
}

/*---------- put text attribute n charater -------------*/
void f_putna (int n ,unsigned char attr)
{
  int i;

  for (i=0;i<n;i++)
     {
     f_puta(attr);
     direct_video.cursor_pos=direct_video.cursor_pos+2;
     if (direct_video.cursor_pos > 3998)
        direct_video.cursor_pos =0;
     }
}

/*----------- write a character on the position ------------*/
void f_putc (unsigned char c)
{
        fast_putc(c);
   direct_video.cursor_pos=direct_video.cursor_pos+2;
   if(direct_video.cursor_pos > 3998)
      direct_video.cursor_pos =0;
}

/*----------- write string on the position --------------*/
void f_puts(unsigned char  *str)
{

  while(*str)
     {
     switch(*str)
        {
         case '\r' :
         case '\n' : break;
         case '\b' : direct_video.cursor_pos=direct_video.cursor_pos-2;
                     break;
         default   : f_putc(*str);
                     break;
```

รูปที่ 7 (ต่อ)

```
        }
      str++;
      }
}

/*----------- get a character on the position ------------*/
unsigned  f_getc (void)
{
   unsigned c;

       c = fast_getc ();
   direct_video.cursor_pos=direct_video.cursor_pos+2;
   if(direct_video.cursor_pos > 3998)
      direct_video.cursor_pos =0;
   return (c);
}
```

รูปที่ 8 ไฟล์ VIDEOA.ASM

```
;---------------------- videoa.asm --------------------
      dosseg
      .model small

SCREEN_DATA struc
   cursorpos dw    0          ; cursor position
   stport    dw    0          ; CRT I/O port
   attribute db    0          ; attribute of char
SCREEN_DATA ends

      .data
      public _direct_video
      _direct_video  SCREEN_DATA <>
      extrn  _screenseg    : word

      .code

;----------------------------
; fast put character to screen
; fast_putc (char c);
;----------------------------
      public _fast_putc
_fast_putc   proc near
      push   bp
      mov    bp,sp
      push   es
      push   di
      mov    es,_screenseg
      mov    di,_direct_video.cursorpos
      mov    dx,word ptr _direct_video.stport
      mov    ch,byte ptr _direct_video.attribute
      mov    cl,byte ptr [bp+4]
putc0:
      cmp    dx,03bah
      jz     monoc
      in     al,dx
      test   al,1
      jnz    putc0
putc1:
      in     al,dx
      test   al,1
      jz     putc1
monoc:
      mov    ax,cx
      stosw
      pop    di
      pop    es
      pop    bp
      ret
_fast_putc endp

;----------------------------------------
; fast put attribute of character to screen
; fast_puta (unsigned char attribute);
;----------------------------------------
      public _fast_puta
_fast_puta   proc near
      push   bp
      mov    bp,sp
      push   es
      push   di
      mov    es,_screenseg
      mov    di,_direct_video.cursorpos
      mov    dx,word ptr _direct_video.stport
```

รูปที่ 8 (ต่อ)

```
        mov     cl,byte ptr [bp+4]
        inc     di
puta0:
        cmp     dx,03bah
        jz      monoa
        in      al,dx
        test    al,1
        jnz     puta0
puta1:
        in      al,dx
        test    al,1
        jz      puta1
monoa:
        mov     al,cl
        stosb
        pop     di
        pop     es
        pop     bp
        ret
_fast_puta endp

;-----------------------------------------------
;get character and attribute from screen
;byte high is an chararter code
;byte low is  an attribute
;int fast_getc (void)
;-----------------------------------------------
        public _fast_getc
_fast_getc   proc near
        push    ds
        push    si
        mov     cx,_screenseg
        mov     si,_direct_video.cursorpos
        mov     dx,word ptr _direct_video.stport
        mov     ds,cx
getc0:
        cmp     dx,03bah
        jz      monog
        in      al,dx
        test    al,1
        jnz     getc0
getc1:
        in      al,dx
        test    al,1
        jz      getc1
monog:
        lodsw
        pop     si
        pop     ds
        ret
_fast_getc endp

end
```

รูปที่ 9 ไฟล์ CLOCK.C ฟังก์ชันป๊อบอัพ ตัวอย่าง

```
/*------------------- clock.c -----------------*/
/*        Pop-Up function example               */

#include <dos.h>
#include <time.h>
#include "tsr.h"
#include "video.h"
#pragma check_stack (off)

#define Y 10          /* where the clock display */
#define X 30

/*---------- varible to control TSR program --------------*/
extern int terminating, suspending;

/* --------- function called when TSR is popped up ---------- */
void popup()
{
   int c, v;
   char str_video[42];
   char buffer [9];

   fast_display ();
   textattr (RevBlinkHigh);
```

รูปที่ 9 (ต่อ)

```
    /* ------ save thr video memory -----*/
    f_gotoxy (X, Y);
    for (v = 0; v < 21; v++)
       {
       c = f_getc ();
       str_video[v*2]   = c;
       str_video[v*2+1] = c >> 8;
       }
    while (!(keyhit()))
       {
       f_gotoxy (X, Y);
       _strtime(buffer);
       f_puts (buffer);
       f_puts ("  ");
       _strdate (buffer);
       f_puts (buffer);
       }

    /* ----- restore the video memory ----- */
    f_gotoxy (X, Y);
    for (v = 0, v < 21; v++)
       {
       textattr (str_video[v*2+1]);
       f_putc ( str_video[v*2]);
       }
    terminating = ((c = get_char()) == 'u');
    suspending = (c == 's');
}
```

รูปที่ 10 การคอมพายล์และลิงค์ไฟล์ทั้งหมดเข้าด้วยกันด้วย MASM (เวอร์ชัน 5.0) และไมโครซอฟต์ซี (CL)

```
Compile and Link
1. Compile assembly program with MASM Ver. 5.0

   C:\>masm -Ml tsrasm;
   Microsoft (R) Macro Assembler Version 5.00
   Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

   51320 + 299096 Bytes symbol space free

        0 Warning Errors
        0 Severe  Errors

   C:\>masm -Ml videoa;
   Microsoft (R) Macro Assembler Version 5.00
   Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

   50960 + 299456 Bytes symbol space free

        0 Warning Errors
        0 Severe  Errors

   The results are  in .OBJ file  TSRASM.OBJ  VIDEOA.OBJ

2. Compile and link with Microsoft C Ver. 5.0 (or later or Quick C)

   C:\>cl -AS tsr.c resident.c videoc.c clock.c tsrasm.obj videoa.obj
   Microsoft (R) C Optimizing Compiler Version 5.00
   Copyright (c) Microsoft Corp 1984, 1985, 1986, 1987. All rights reserved.

   tsr.c
   resident.c
   videoc.c
   clock.c

   Microsoft (R) Overlay Linker  Version 3.60
   Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

   Object Modules [.OBJ]: TSR.OBJ +
   Object Modules [.OBJ]: RESIDENT.OBJ +
   Object Modules [.OBJ]: VIDEOC.OBJ +
   Object Modules [.OBJ]: CLOCK.OBJ +
   Object Modules [.OBJ]: TSRASM.OBJ +
   Object Modules [.OBJ]: VIDEOA.OBJ
   Run File [TSR.EXE]: TSR.EXE /NOI
   List File [NUL.MAP]: NUL
   Libraries [.LIB]: ;

   The out-put file is TSR.EXE.
```